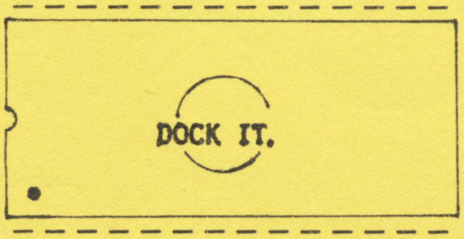
DOCK IT

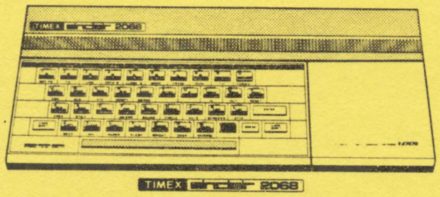
$17.95



EPROM PROGRAMMER MANUAL

for the

TS2068 Computer



with ROM disassembly

by

Bob Orrfelt

GESSO
PRODUCTS

2nd printing Oct. 85

Copyright ⓒ 1985 by Bob Orrfelt. All rights reserved.

Section 1.

INTRODUCTION

The GESSO EPROM Programmer and EPROM ZAPPER software are designed to produce instant loading software for the TS2068 computer. The hardware and software work together to put BASIC programs, machine code, and data on EPROMs for the TS2068 cartridge port (DOCK).

The EPROM Programmer programs 2764 (8K) or 27128 (16K) EPROMs. The unit fits in the cartridge port. An on board voltage regulator uses a 24vac power supply from a TS2040 printer. For the programmer to work a modification to the TS2068 is required (see Section 4).

The EPROM ZAPPER software is on EPROM for instant loading. Programming software is below BASIC and the ZEUS assembler is in high RAM. After loading the TS2068 is returned to the normal operating mode and the EPROM ZAPPER EPROM can be removed from the computer.

A ZEUS ASSEMBLER is included as part of the system. It is used to alter the EPROM ZAPPER software when necessary and to produce headers and code for the cartridge EPROM.

A TAPE HEADER READER EPROM is included to get size and auto start data from tape programs.

A GESSO CARTRIDGE BOARD with a ZIF (zero insertion force) socket is also included to complete the hardware needed.

This manual DOCK IT contains operation, theory, hardware, and a TS2068 ROM disassembly as an aid in programming EPROMs.

Once a program is in suitable form on tape or on another EPROM, the process of putting it on an EPROM is quite simple. BASIC programs are run directly out of the DOCK - one line at a time. Machine code programs may run directly but if RAM in the same Chunk is required, it needs to be moved to RAM. In such a case a "MOVER" is needed and this may not be quite so simple.

The ZEUS Assembler provides a solution. There are two parts: a Z80 assembler and a monitor. A short assembly language program can be written to move code for the EPROM. The monitor can be used to review code to be put on EPROM. The monitor also permits poking RAM one byte at a time.

You will find the ZEUS assembler requires a # before hex numbers but the monitor does not. An exception is that the monitor A command for converting decimal/hex does require the preceding #.

The ZEUS assembler normally puts the source file at #8000. This is the RAM address for the EPROM source program. Use N#C000 to put the source file where it will do no harm.

The TAPE HEADER READER program will read and print if you wish the header information from cassette tapes. Here is an example of a look at the ZEUS assembler.

```
ABBREVIATIONS:

 L: LENGTH
 S: START ADDR
AN: ARRAY NAME
PL: PROG LENGTH
AS: AUTOSTART AT LINE
TL: TOTAL LENGTH (PROG+VARS)

NAME: ZEUS          TYPE: PROGRAM
TL:683  AS:1  PL:683

NAME: ZEUS          TYPE: BYTES
L:7936  S:57344
```

Figure 1-1. Tape Headers

In the above example only the machine code (BYTES) are really needed for an EPROM cartridge. However, a look at the code shows that it must reside in RAM. Therefore a mover is required to make it work.

Section 2.

OPERATION

The GESSO EPROM Programmer has two switches, a zif (zero insertion force) socket, a 24 vac power supply jack, and two LED indicators. The unit fits in the cartridge port of the TS2068 computer. The programmer is designed to program 2764 (8k) or 27128 (16k) EPROMs. The programmer also functions as an EPROM reader.

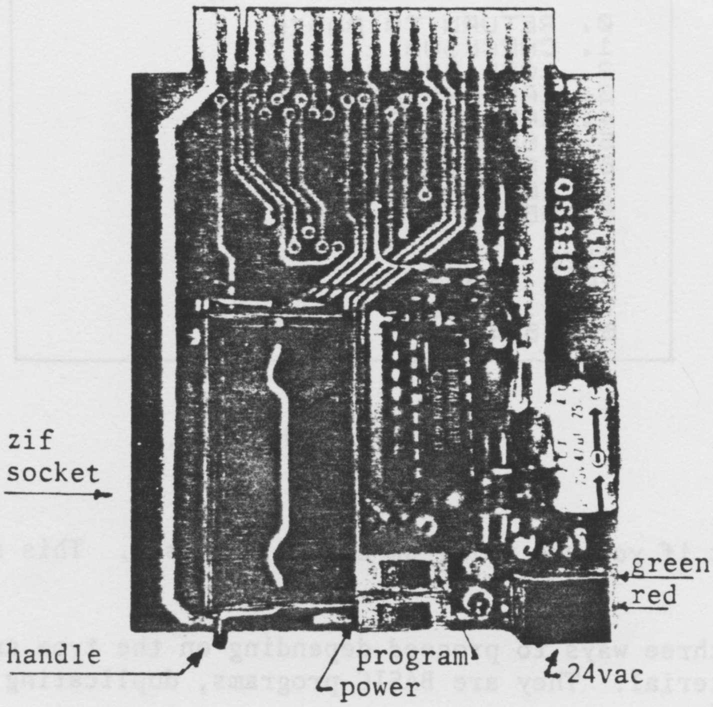


Figure 2-1. Controls

Always turn the cartridge board power supply switch to off before inserting or removing the board.

Always remove the cartridge board when changing EPROMs.

TO START:

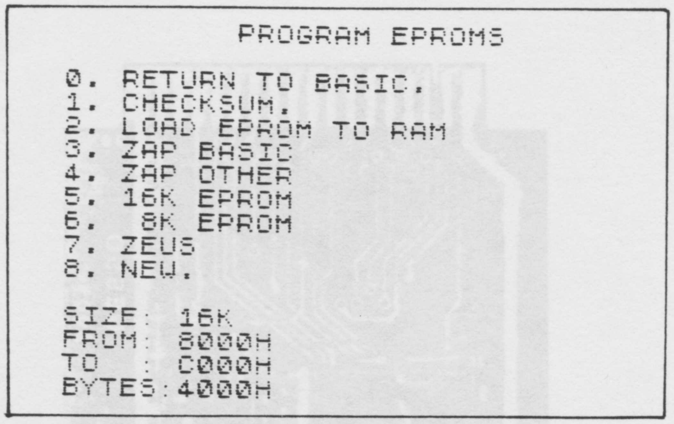
1. Have the program you want on EPROM ready, either on tape or on another EPROM.

2. Insert the EPROM ZAPPER in the DOCK and turn computer power. If the program loads properly, you get the normal sign on message except the screen is reversed. The zapper program is in RAM below BASIC and ZEUS is in high RAM.

3. Turn the cartridge switch to off and remove the cartridge.

4. Next we need the program that's going on EPROM. If your program is on tape "merge" the BASIC program or load code into RAM between "32768 and 49151.

5. Enter PRINT USR 24333 and the menu will appear.



```
          PROGRAM EPROMS

  0. RETURN TO BASIC.
  1. CHECKSUM.
  2. LOAD EPROM TO RAM
  3. ZAP BASIC
  4. ZAP OTHER
  5. 16K EPROM
  6.  8K EPROM
  7. ZEUS
  8. NEW.

  SIZE: 16K
  FROM: 8000H
  TO  : C000H
  BYTES:4000H
```

Figure 2-2. The menu.

6. Next if you are using 8K EPROMs, press 6. This sets size and bytes.

There are three ways to proceed depending on the type and source of the program material. They are BASIC programs, duplicating an EPROM, and machine code.

BASIC PROGRAMS

See Section 3 of this manual for restrictions on BASIC programs. For a simple BASIC program that follows the rules, proceed as follows:

1. Raise the ZIF lever on the EPROM Programmer to vertical, insert a blank EPROM, close the lever.

2. Turn both programmer switches to off (right) and plug the programmer into the TS2068 cartridge slot.

3. Slide the programmer power switch (nearest to the front) to on (left). The red LED will come on dimly.

4. Press 1 for checksum. 8K blank = E000 or 16K blank = C000.

5. Press 3. Step through the messages to: PLUG IN 24 V AC. (You already have the EPROM in). Follow the instructions. The green LED will blink as each byte is programmed. The header is put on automatically. Programming time will depend on the length of the BASIC program.

6. Press 1. You may wish to note the checksum for future referance.

7. Press 8. This tests that the EPROM is properly programmed.

DUPLICATING EPROMS

To duplicate an EPROM proceed as follows:

1. Insert the master EPROM in a cartridge board. (Can be the EPROM Programmer board).

2. Turn the cartridge board power to off and insert in the cartridge slot.

3. Turn the cartridge board power on.

4. Press 1 for checksum.

5. Press 2. You already have the master in place so just step through the messages.

6. Turn the cartridge board power to off and remove the board.

7. Insert a blank EPROM in the EPROM programmer board.

8. With the switches off, insert the board.

9. Press 1 for checksum. 8K blank = E000 or 16K blank = C000.

10. Press 4. Step through to PLUG IN 24 V AC and follow the messages. An 8k takes about 7 minutes and a 16K about 15 minutes.

11. Press 1 for checksum.

12. Press 8 to test if you wish.

MACHINE CODE

If you have loaded a machine code from tape it must have the correct header at 8000 hex (32768). See the next section for details. Once you have the correct code starting at 8000 hex go to step 7 above as for duplication.

You can LIST or LLIST a BASIC AROS by:

ENTER:

```
  POKE 23750,0
```

this turns off the BASIC AROS flag.

Now ENTER on on line:

```
  POKE 23635,8:POKE 23636,128:OUT 244,48:LIST
```

(or LLIST).

this pokes the starting address of the BASIC program into PROG, switches in the DOCK, and lists the program.

THEORY

There are three formats reconized by the TS2068 on initialization: LROS, BASIC AROS, and Machine Code AROS. BASIC AROS cartridges can also contain machine code, BASIC variables, etc.

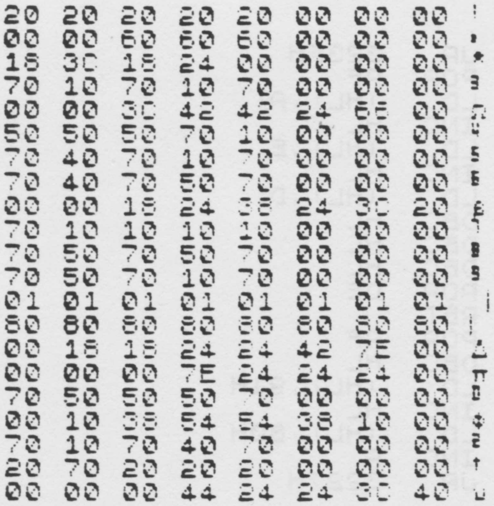
LROS

An LROS replaces the operating system and takes over all functions of the computer. An example of an LROS is the SPECTRUM emulator. During the initialization process the cartridge is switched in and the second byte is read. If it is an 01 the fifth byte is used to switch in the proper chunks. A jump is then made to the address in the third and fourth bytes. From now on control is passed to the cartridge EPROM.

Making a SPECTRUM EPROM gives the opportunity to correct some of the errors in the original version. Example 1 on the next page shows my version with modifications.

Addresses 0001-0004 must be replaced with the LROS header. The starting address in the header is therefore to an unused SPECTRUM address (3870h). This address contains the original code starting at 0000 and ends with a jump to the normal initialization address for the SPECTRUM.

An alternate USR character set was added at 3CFFH.



```
20  20  20  20  20  00  00  00
00  00  60  60  60  00  00  00
18  3C  18  24  00  00  00  00
70  10  70  10  70  00  00  00
00  00  3C  42  42  24  66  00
50  50  50  70  10  00  00  00
70  40  70  10  70  00  00  00
70  40  70  50  70  00  00  00
00  00  18  24  38  24  3C  20
70  10  10  10  10  00  00  00
70  50  70  50  70  00  00  00
70  50  70  10  70  00  00  00
01  01  01  01  01  01  01  01
80  80  80  80  80  80  80  80
00  18  18  24  24  42  7E  00
00  00  00  7E  24  24  24  00
70  50  50  50  70  00  00  00
00  10  38  54  54  38  10  00
70  10  70  40  70  00  00  00
20  70  20  20  20  00  00  00
00  00  00  44  24  24  3C  40
```

Figure 3-1. Mod B USR Character Set.

BASIC AROS

Of the three formats this is the most interesting. A BASIC program which conforms to a few limitations will run directly with no problems. In addition

(cont. 3-3)

The following are changes to the Spectrum code designated as Mod. B. The addresses and codes are given in hex. Thanks to Dr. Ian Logan Dr. Frank O'Hara for identifying the "bugs".

```
1.  0001 01  LROS
    0002 70
             Starting address 3870
    0003 38
    0004 FC  Chunks 1 and 2
    0005 42  Mod. #  (B)

2.  006D 28  JR Z  was JR NZ

3.  0A33 19  was 18  CHR$ 8 (backspace) error line 0 etc.

4.  1204 FF
             address of new USR graphic characters (3CFF)
    1205 3C

5.  1539     changed logo
                                          1539 Sinclair Spectru
                                          1549 m Mod. B

6.  257D C9  was C3 SCREEN$ error

7.  3200 DB  was E1  1/2-.5 error

8.  3032 F5          PUSH AF              INT -65536 etc. error
    3033 3C          INC  A
    3034 B3          OR   E
    3035 B2          OR   D
    3036 C22532      JP   NZ,3225H
    3039 C33032      JP   3230H

    3223 181A        JR   323FH
    3225 F1          POP  AF
    3226 77          LD   (HL),A
    3227 23          INC  HL
    3228 73          LD   (HL),E
    3229 23          INC  HL
    322A 72          LD   (HL),D
    322B 2B          DEC  HL
    322C 2B          DEC  HL
    322D 2B          DEC  HL
    322E D1          POP  DE
    322F C9          RET
    3230 F1          POP  AF
    3231 2B          DEC  HL
    3232 3691        LD   (HL),91H
    3234 23          INC  HL
    3235 3680        LD   (HL),80H
    3237 3C          INC  A
    3238 18ED        JR   3227H

9.  3870 F3          DI                   Boot from address 0000
    3871 AF          XOR  A
    3872 11FFFF      LD   DE,FFFFH
    3875 C3CB11      JP   11CBH
```

Example 1. SPECTRUM EPROM Mod. B.

machine code, variables, USR defined graphics, etc. can be added to a BASIC AROS. But first let's compare formats.

CARTRIDGE FORMATS

LROS		BASIC AROS		MACHINE CODE AROS	
Address	Code	Address	Code	Address	Code
0000		8000h	01	8000	02
0001	01	8001	02	8001	02
0002/3	jump addr.	8002/3	goto addr.	8002/3	jump addr.
0004	chunks	8004	chunks	8004	chunks
0005		8005	auto (01)	8005	auto (01)
		8006/7	# of vars.	8006/7	# of vars.
Note: For # of (machine code) variables add 21 (15h).					

If it is an AROS the code is copied from the EPROM into a system configuration table (see inside back cover). The chunk(s) in the DOCK header code is designated by binary 0, i.e. FEh=1111 1110 or chunk 0. In machine code the chunk(s) is designated by binary 1, i.e. 01=0000 0001 or chunk 0. (also inside back cover).

If a BASIC program conforms to the limitations listed on the next page, the ZAP BASIC command will add the header and copy the BASIC program onto EPROM. If it is necessary to be more creative, an understanding of how a BASIC AROS works is needed.

When the computer is turned on or after NEW is executed, the system is initialized. That is, memory RAM is set to zeros, the system variables are copied from ROM to RAM, the extension ROM bank is switched in and the bank switching is copied into RAM. Next the DOCK bank is switched in (chunk 4) and the first 8 bytes are copied into RAM (the system configuration table). If these bytes show a BASIC AROS is present and the auto start is indicated, then the BASIC program in the cartridge is run. If the auto start byte is zero, then the home bank is set to home ROM/RAM.

In the auto start mode, the first line indicated by the low/high starting adress is copied into the ARSBUF, the cartridge switched out, and the line is executed. Next, the cartridge is switched in, the next line copied, the cartridge switched out, and the next line is run. For READ statements the entire DATA line is also copied into RAM. Control of all this is through bank switching.

LIMITATIONS on DOCK BASIC programs:

1. BASIC variables must be initialized from a BASIC line, i.e. LET A$="Hi".
2. DEF FN and FN functions cannot be used in a DOCK BASIC program. Replace all FN's with the function and delete the DEF FN.
3. No 64 column video display (dual video mode).
4. No calculated GO TO or GOSUB numbers. i.e. GO TO (1000+n*10).
5. The byte following a BASIC program must be 128 or greater. Leaving the following byte at 255 satisfies this requirement.
6. FOR/NEXT cannot pass the limit:

   ```
   For A=B TO 10
   LET B=12
   NEXT A         won't work!
   ```

7. There is a bug in the USR subroutine so that only some of the addresses are valid. See the table below.

VALID DOCK USR ADDRESSES

8k EPROM		16k EPROM	
32768-33279	(8000h-81FFh)	32768-33791	(8000h-83FFh)
34816-35327	(8800h-89FFh)	34816-35839	(8800h-8BFFh)
36864-37375	(9000h-91FFh)	36864-37887	(9000h-93FFh)
38912-39423	(9800h-99FFh)	38912-39935	(9800h-9BFFh)
41472-41983	(A200h-A3FFh)	40960-41983	(A000h-A3FFh)
43520-44031	(AA00h-ABFFh)	43008-44031	(A800h-ABFFh)
45568-46079	(B200h-B3FFh)	45056-46079	(B000h-B3FFh)
47616-48127	(BA00h-BBFFh)	47104-48127	(B800h-BBFFh)
50176-50687	(C400h-C5FFh)	50176-51199	(C400h-C7FFh)
52224-52735	(CC00h-CDFFh)	52224-53247	(CC00h-CFFFh)
54272-54783	(D400h-D5FFh)	54272-55295	(D400h-D7FFh)
56320-56831	(DC00h-DDFFh)	56320-57343	(DC00h-DFFFh)
58880-59391	(E600h-E7FFh)	58368-59391	(E400h-E7FFh)
60928-61439	(EE00h-EFFFh)	60416-61439	(EC00h-EFFFh)
62976-63487	(F600h-F7FFh)	62464-63487	(F400h-F7FFh)
65024-65535	(FE00h-FFFFh)	64512-65535	(FC00h-FFFFh)

INs, OUTs, and MEMORY

Before getting into memory, chunks, and bank switching it is a good idea to look at I/O controls in general. The Z80 CPU provides for 256 "ports" in and 256 "ports" out. The port in and the port out can be the same or different depending on the decoding. From a hardware viewpoint, the Z80 CPU address lines A0-A7 and $\overline{\text{IOREQ}}$ select the port; $\overline{\text{WR}}$ selects in and $\overline{\text{RD}}$ selects out. This decoding occurs external to the Z80 CPU. In the TS2068 some of the ports are decoded by the SCLA as shown in figure 3-2. (The SCLA performs many functions. Only a few are shown in the figure).

There are five ports decoded by the SCLA and each is quite different in the way it behaves. Remember IN is into the Z80 CPU and OUT is into the SCLA. A register stores 8 bits (usually). Here are the five ports:

FFh (255 decimal): the Enhancement Port. This is an eight bit register which can be read or written to. The lower six bits control the 64 column mode including paper and ink colors. The seventh bit (D6) controls the interrupt generated by the keyboard logic. The eighth bit (D7) with port F4h (244 decimal) selects memory.

FEh (254 decimal): Keyboard/Tape/Border. This port has two registers, one for IN and one for OUT. The OUT register stores the output to the speaker and tape and the boarder color. The IN register stores the tape and keyboard inputs. The keyboard uses additional address lines to read the keyboard.

F5h (245 decimal) and F6h (246 decimal): Sound Chip. The sound chip is external to the SCLA and contains a number of registers. The SCLA decodes the F5h port to load a register address into the sound chip and port F6h is decoded to load data into the selected register. The sound chip contains a register called Port A. This register holds the joystick data. Port A also holds the state of a pin on the rear edge connector (P1) called IOA5. Port A is bidirectional and can be used to send data in either direction.

F4h (244 decimal): Dock Horizontal Select. Together with bit 7 of port FFh, this register stores the "chuncks" (8k) of memory selected from the home bank, extension bank, or the dock (cartridge slot).

All other ports must be decoded external to the TS2068 if they are to be used and must not conflict with ports already in use. The 2040 printer uses up 64 ports because the address lines are not fully decoded. Some other printer interfaces use port 7Fh (127 decimal).

Ports can be read or written to by the BASIC commands IN and OUT. For example: OUT 254,3 will temporarily set the border color to 3. Try this:

```
10 FOR n=0 TO 239
20 OUT 254,16+n
30 PAUSE 1
40 OUT 254,0
50 PAUSE 1
60 NEXT n
```

You figure it out.

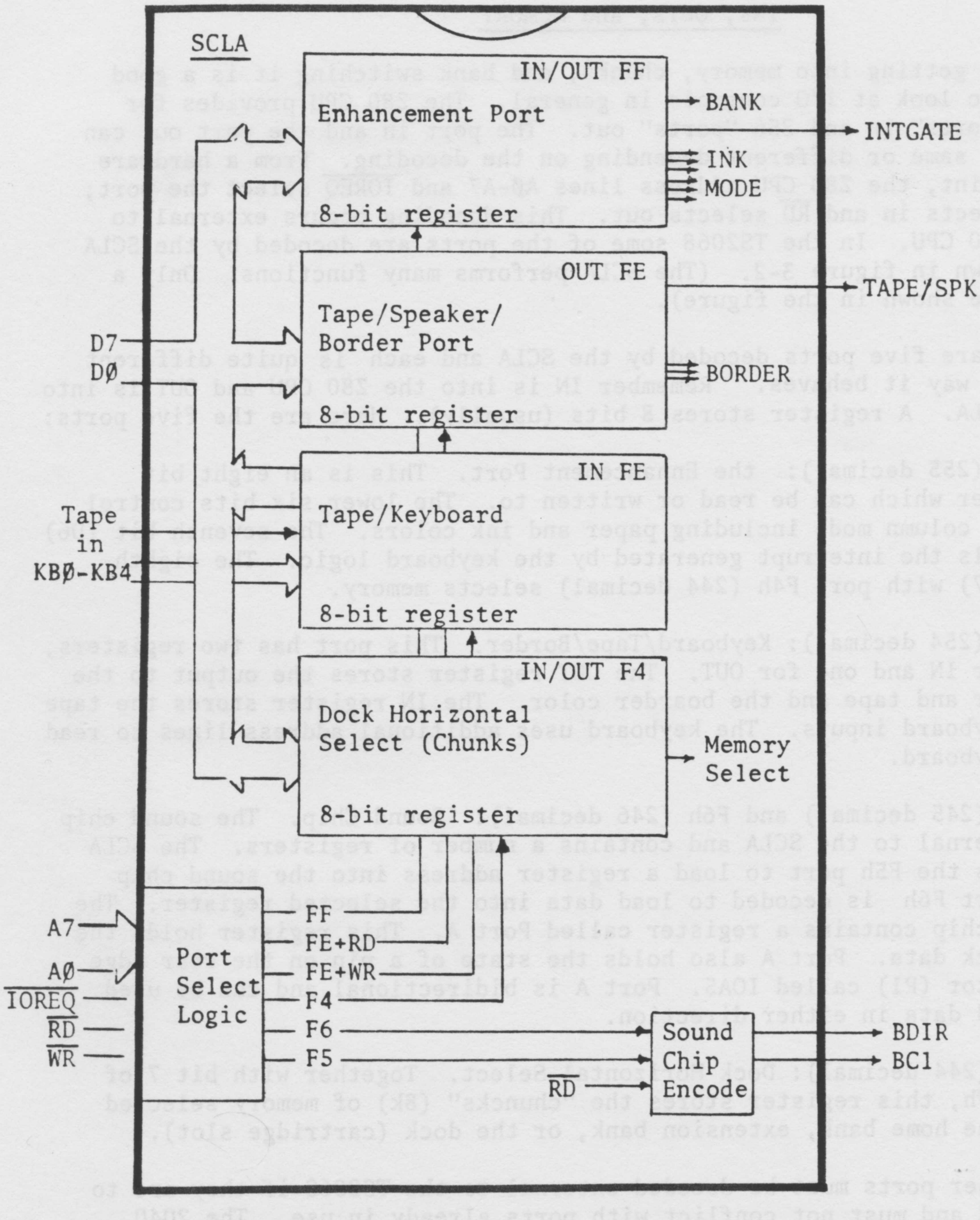


Figure 3-2. SCLA Ports.

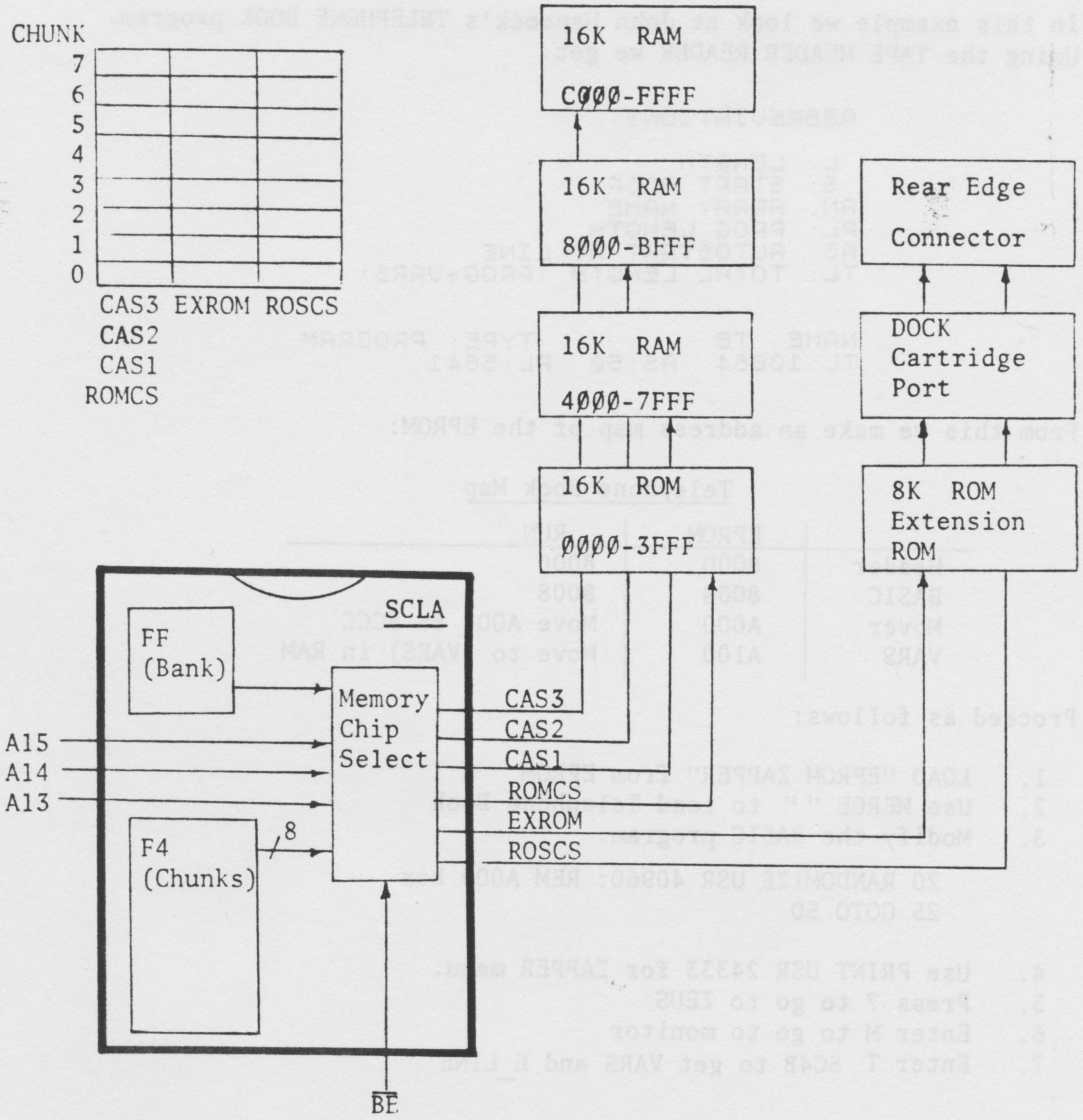


Figure 3-3. Internal Bank Switching.

Figure 3-3 shows the internal bank switching. The $\overline{BE}$ line is from the rear edge connector for use by external banks. Its function is to turn off all internal memory.

In this example we look at John Hancock's TELEPHONE BOOK program. Using the TAPE HEADER READER we get:

```
ABBREVIATIONS:

 L: LENGTH
 S: START ADDR
AN: ARRAY NAME
PL: PROG LENGTH
AS: AUTOSTART AT LINE
TL: TOTAL LENGTH (PROG+VARS)

NAME: TB                TYPE: PROGRAM
TL:10064  AS:50  PL:5641
```

From this we make an address map of the EPROM:

Telephone Book Map

	EPROM	RUN
Header	8000	8000
BASIC	8008	8008
Mover	A000	Move A00E to 5CCC
VARS	A100	Move to (VARS) in RAM

Proceed as follows:

1. LOAD "EPROM ZAPPER" from EPROM
2. Use MERGE "" to load Telephone Book
3. Modify the BASIC program:

```
20 RANDOMIZE USR 40960: REM A000 hex
25 GOTO 50
```

4. Use PRINT USR 24333 for ZAPPER menu.
5. Press 7 to go to ZEUS
6. Enter M to go to monitor
7. Enter T 5C4B to get VARS and E_LINE

```
M
Monitor
T5C4B                VARS                  E_LINE

5C4B 5F 7E 00 00 40 68 4F 68
5C53 56 68 B5 8F 55 68 A7 8F
5C5B AE 8F B4 8F CF 00 B6 8F
5C63 B6 8F B6 8F 2D 92 5C 02
5C6B 02 00 00 00 00 00 00 00
5C73 00 9E 19 00 00 37 53 00
5C7B 58 FF 00 00 10 14 5B 21
5C83 17 00 50 E0 50 21 08 21
```

8. E_LINE-VARS+1=1148 hex bytes (+1 for end marker)
9. Enter C 7E5F A100 1148 to copy vars to RAM

Example 2. BASIC + BASIC Variables

10. Enter Z to goto ZEUS
11. Enter N #C000 and then I. Type in the mover as follows:

```
00010     ORG #A000
00020    LD   HL,#A00E    move mover
00030    LD   DE,#5CCC    to RAM
00040    LD   BC,#0030
00050    LDIR
00060    JP   #5CCC       mover
00070    LD   A,#C0
00080    OUT  (#F4),A     chunks 6&7
00090    LD   HL,(#5C4B)  VARS
00100    EX   DE,HL
00110    LD   HL,#E100    from
00120    LD   BC,#1148     to
00130    LDIR
00140    EX   DE,HL
00150    LD   (#5C59),HL   E_LINE
00160    CALL #1342
00170    XOR A
00180    OUT  (#F4),A     DOCK out
00190    RET
```

12. Enter A to assemble at A000.
13. Next to alter the "ZAP OTHER" parameters
    Enter M and then M 5F00. Make changes as follows:

```
M
Monitor          after
M5F00
5F00 00 5F00 00
                        SIZE:
5F01 40 5F01 40

5F02 00 5F02 00
                        BYTES:
5F03 40 5F03 20

5F04 00 5F04 00
                        TO:
5F05 C0 5F05 E0

5F06 00 5F06 00
                        FROM:
5F07 80 5F07 A0

5F08 00 5F08 00.
```

14. Things are now set up for making the EPROM. You can verify the code using the monitor:

    T 5F00   for from, two, bytes
    T A000   mover
    T A100   BASIC variables

15. Enter Z then Q to get back to the ZAPPER menu.
16. Using a blank EPROM zap BASIC. (Remember the BASIC program is still there).
17. Now ZAP OTHER. This puts the mover and BASIC VARS into EPROM.
18. Press 8 for NEW and test the program.

Example 2. BASIC + BASIC Variables (cont.)

A look at the tape headers shows VU-FILE (tm PSION) to have both BASIC and machine code programs. Further the machine code is to start at 28288 (6E80h). This means the code extends from chunk 3 to chunk 4 and needs to be in RAM. It was decided to use a 16k EPROM in chunks 6 and 7. Here is a map:

VU-FILE MAP

	RAM	EPROM	RUN
Header	8000	C000	
BASIC	8008	C008	BASIC in chunks 6 and 7
Mover	9C00	DC00	Valid USR address (5632d)
Code	A300	E300	Move to 6E80 (28288d) in RAM

Proceed as follows:

1. LOAD "EPROM ZAPPER" from EPROM
2. Use MERGE "" to get vfile from tape
3. Use LOAD "c" CODE 41936,7216 (41396=A3D0h).
4. Modify the BASIC program:
   10 RANDOMIZE USR 56320
   and delete LOAD "" SCREEN$: LOAD "" CODE from line 50.
5. Use PRINT USR 24333 for ZAPPER menu.
6. Press 7 to goto ZEUS
7. Enter M to goto monitor.
8. Enter M 5EF7. change code as follows:

```
5EF7 01 5EF7 01   BASIC
5EF8 02 5EF8 02   AROS
5EF9 08 5EF9 08   START
5EFA 80 5EFA C0   ADDRESS
5EFB CF 5EFB 3F   CHUNKS
5EFC 01 5EFC 01   AUTO
5EFD 15 5EFD 15   MVARS
5EFE 00 5EFE 00     "
5EFF 00 5EFF 00   NOP
5F00 00 5F00 00
5F01 40 5F01 40   SIZE:
5F02 00 5F02 00
5F03 40 5F03 24   BYTES:
5F04 00 5F04 00
5F05 C0 5F05 DC   TO:
5F06 00 5F06 00
5F07 80 5F07 9C   FROM:
5F08 00 5F08 00.
```

Example 3. BASIC + Code in RAM.

9. Enter Z to goto ZEUS
10. Enter N#C000 and then I to write the mover source program: Comments are not necessary.

```
00010    ORG #9C00
00020   LD   HL,#E3D0   ;FROM :
00030   LD   DE,#6E80   ;TO   :
00040   LD   BC,#1C30  ;BYTES:
00050   LDIR
00060   XOR A
00070   LD    (#5C41),A ;MODE K/L
00080   LD   HL,#5C3B
00090   SET  3,(HL)      ;MODE L
00100   LD   HL,#5C92   ;fp STACK
00110   LD    (#5C68),HL
00120   RET
```

11. Enter A to assemble source at 9C00h.
12. Things are now setup for making the vu-file EPROM. You may examine the code by entering M for monitor and then:

    T 5EF7 for BASIC header and size etc.

    T 9C00 for mover of code to RAM

    T A3D0 for vu-file machine code

13. Enter Z then Q to get back to the ZAPPER menu.
14. Use a blank EPROM (16k) and zap BASIC.
15. Now zap OTHER. This puts in the mover and vu-file machine code.
16. Use checksum and make a note, mine is 6D7B.
17. Press 8 for NEW and test the program.

Example 3. BASIC + Code in RAM.

The EPROM ZAPPER program is a machine code AROS. There are two independent machine codes: the EPROM programmer and the ZEUS assembler. The EPROM programmer code needs to be moved to a region below BASIC and the BASIC variables. The ZEUS assembler needs to be moved to high memory. Here is the map:

EPROM ZAPPER MAP

	EPROM	RUN
Header	8000	8000
Mover	8008	8008
ZAPPER	8100	5CCC protected low memory
ZEUS	9000	E100 protected high memory

The mover lowers RAMTOP to protect ZEUS, sets the border and ink colors, moves the ZAPPER code, moves the ZEUS code, and then jumps to initialization code in the EXROM. The zapper code has overwritten the system configuration table. Initialization continues as if the DOCK EPROM were not present. The TS2068 is now ready to use as if the computer were just turned on.

The EPROM programmer code is located as follows:

5CCC ASCII messages
5EF7 BASIC EPROM header (8 bytes)
5F00 Size, Bytes, To, From (may be changed)
5F0D Start of program

Here is the header program:

```
00010   ORG  #8000
00020   DEFB #02   ;MACHINE CODE
00030   DEFB #02   ;AROS
00040   DEFB #08   ;START
00050   DEFB #80   ;ADDRESS
00060   DEFB #01   ;AUTO
00070   DEFB #15   ;NO
00080   DEFB #00   ;VARS
00090   LD    HL,#DFFF ;LOWER
00100   LD  (#5C82),HL ;RAMTOP
00110   LD    A,7      ;INK
00120   LD  (#5C48),A
00130   LD  (#5C8D),A
00140   LD    HL,#8100 ;ZAPPER
00150   LD    DE,#5CCC ;TO
00160   LD    BC,#03F0 ;BYTES
00170   LDIR
00180   LD    HL,#9000 ;ZEUS
00190   LD    DE,#E000 ;TO
00200   LD    BC,#1F00 ;BYTES
00210   LDIR
00220   LD    HL,#08F0 ;EXROM
00230   CALL #6815     ;GOTO BANK
```

Example 4. Machine Code AROS

Section 4.

HARDWARE

This section includes the modification to the TS2068 required for operation of the EPROM Programmer, a schematic and parts list of the EPROM Programmer, and information on the GESSO cartridge board.

Modifying the TS2068 DOCK for the EPROM Programmer.

The GESSO DOCK EPROM Programmer requires the Z80 $\overline{\text{WAIT}}$ line connection in order to operate. This modification should only be done by a qualified technician.

A. Remove the seven screws from the bottom of the TS2068.

B. Carefully remove the keyboard connector strip from the conector strip socket.

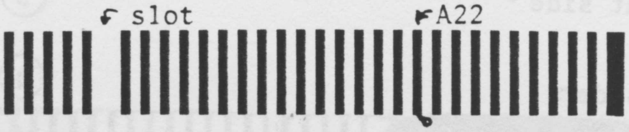
C. Remove the three screws holding the pcb to the bottom of the case.

D. Cut the trace from J4-33 (DOCK) as shown in Figure 4-1.

E. Solder a fine insulated wire to J4-33.

F. Feed this insulated wire through the plated-through hole P1-A22 as shown in Figure 4-1.

G. Solder the end to P1-A22 on the component side. Here is a template:



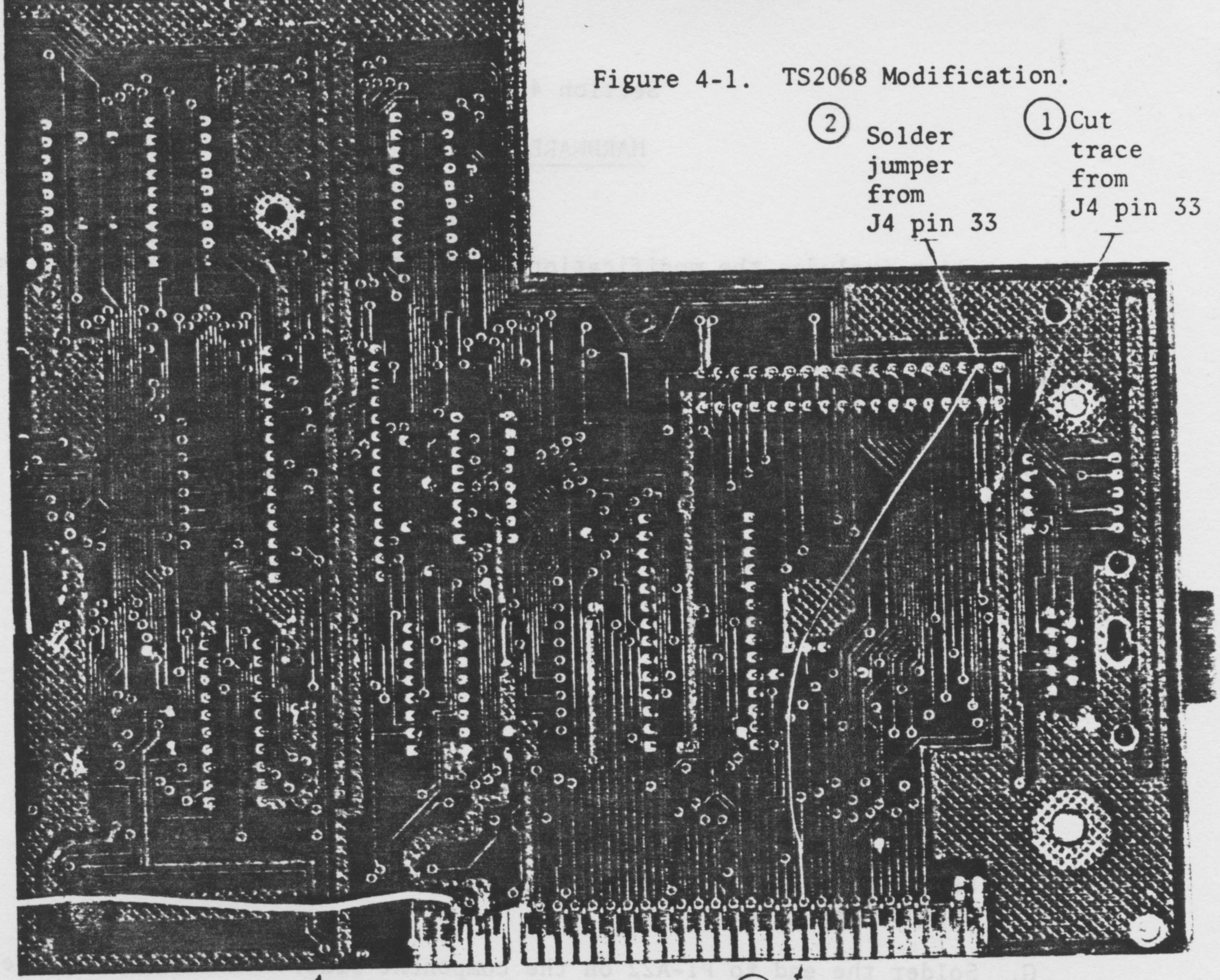
H. Reassemble the TS2068. The three longer screws belong on the rear three holes in the bottom of the case.

THE EPROM PROGRAMMER

The EPROM Programmer schematic is shown in Figure 4-2. A PN2222 is used as a voltage regulator to produce the 21 vdc required for programming 2764's and 27128's. Diode D4 provides 5 vdc to pin 1 of the EPROM for reading the EPROM when the +21 vdc is not present. If the +21 volts is present, the red LED will light brightly but if only the +5 vdc is present the LED will light dimly.

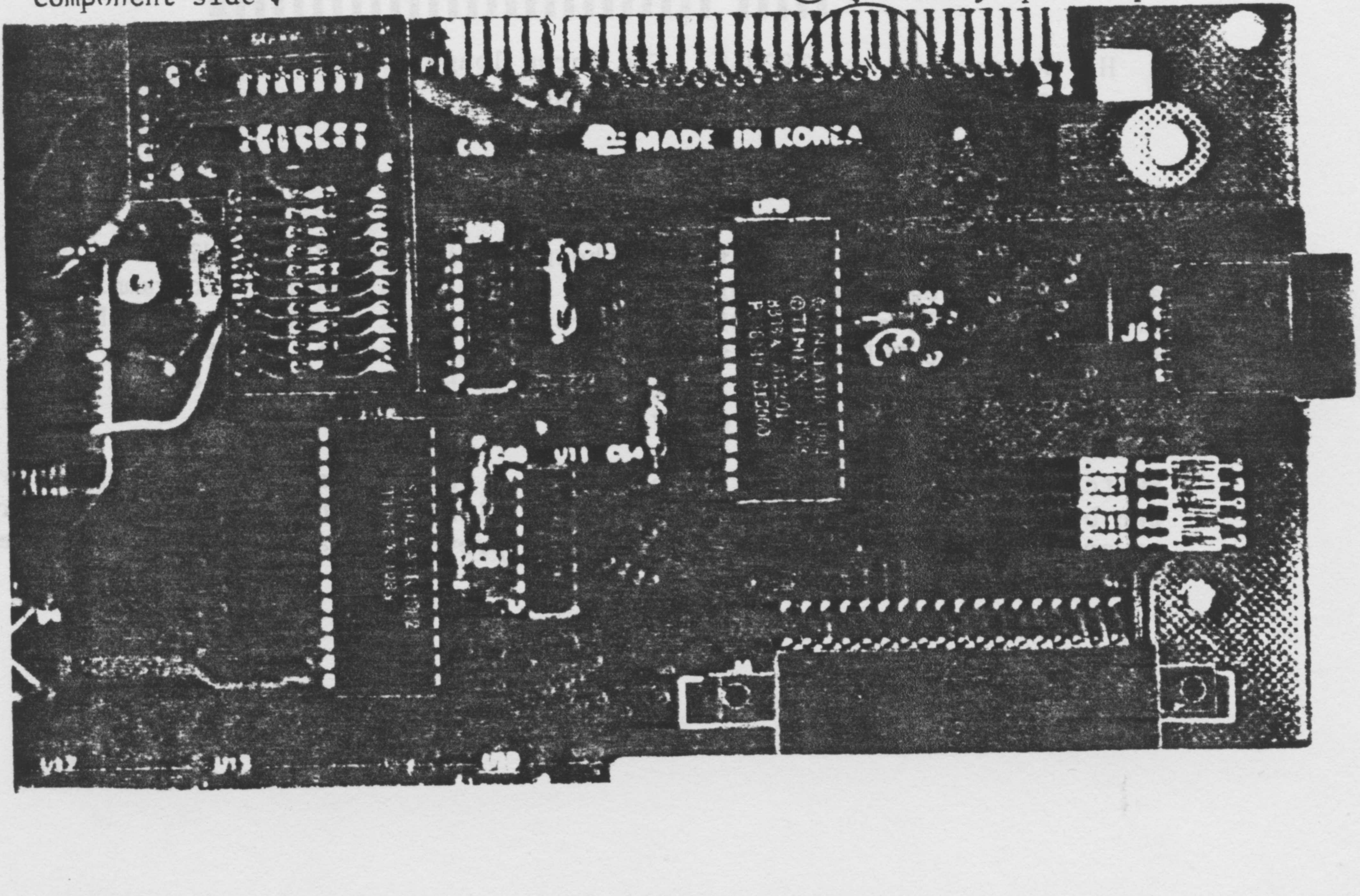
Switch S1 (front) turns on and off the + 5 vdc so that the cartridge board can be removed for loading and unloading without turning off the computer power. Switch S2 (rear) is a lockout to prevent unwanted programming when not in the program mode.

Figure 4-1. TS2068 Modification.


② Solder jumper from J4 pin 33
① Cut trace from J4 pin 33
③ feed jumper through hole

noncomponent side

component side



Diodes D7 and D8 form an and gate for chip select ($\overline{\text{CE}}$) and enables the timing chip (74LS221) in the programming mode. If either $\overline{\text{MREQ}}$ or $\overline{\text{ROSCS}}$ are high (false) the cartridge board is not selected by the computer.

When S2 is in the ON position (left) and the cartridge board is selected, the timer circuit is triggered. C3 and R7 form the 50 ms pulse required for programming each byte. This pulls the $\overline{\text{WAIT}}$ line low. In response the Z80 in the computer freezes the address lines and data lines for the duration of the pulse.

R5 and C6 form a slight delay for voltages to settle. The 74LS221 was selected because it has schmitt trigger inputs so the RC timing will be reproducible. The $\overline{\text{PGM}}$ line then goes low and the byte is programmed.

While the WAIT line is low the green LED D2 turns on. 50 ms is just long enough to cause the LED to blink. This verifies that the programming is proceeding properly.

The Z80 is too fast getting the next byte so special software is required to give the timing capacitors time to recharge.

While changing EPROMS with the computer turned on, the cartridge board should be removed from the DOCK even though the board power is turned off. Noise produced on the address and data lines will probably cause problems for the Z80 and loose data.

The EPROM Programmer parts list is shown in Figure 4-3. The PC board is a modified GESSO cartridge board and is hand wired.

GESSO CARTRIDGE BOARD

Figures 4-4 and 4-5 show the GESSO Cartridge Board. The TS2068 does not include $\overline{\text{MREQ}}$ as part of the $\overline{\text{ROSCS}}$ logic. An $\overline{\text{IOREQ}}$ can produce the $\overline{\text{ROSCS}}$ if the relevent chunk is selected. This means that the $\overline{\text{MREQ}}$ must be used either on the $\overline{\text{OE}}$ pin as shown or for the $\overline{\text{CS}}$ pin as with the EPROM Programmer.

The GESSO Cartridge Board is designed to be adaptable for many ROMs, EPROMs, and RAM chips. Consult the pin out of the device you want and rewire as needed.

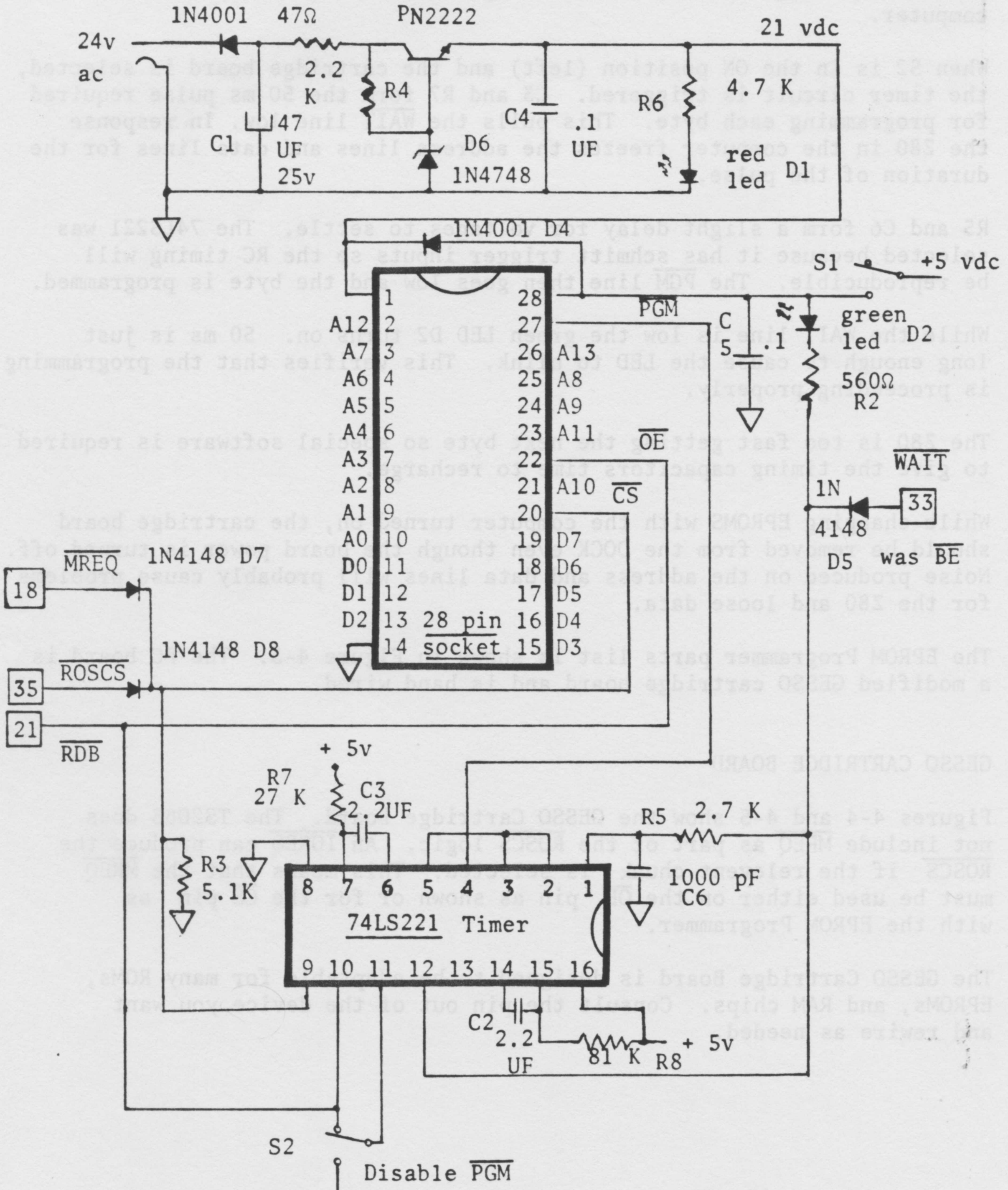


Figure 4-2. EPROM Programmer Schematic

PC Board
28 Pin socket
16 Pin socket
2.1 mm jack
U2-74LS221
Q1-PN2222
D1-red LED
D2-green LED
D3-1N4001
D4-1N4001
D5-1N4148
D6-1N4748
D7-1N4148
D8-1N4148
S1-SPST switch
S2-SPST switch
C1-47 uf 25 vdc
C2-2.2 uf
C3-2.2 uf
C4-.1 uf
C5-.1 uf C6 -.001 uf
R1-47 ohm
R2-560 ohm
R3-5.1 K
R4-2.2 K
R5-2.7 K
R6-4.7 K
R7-27 K
R8-81 K

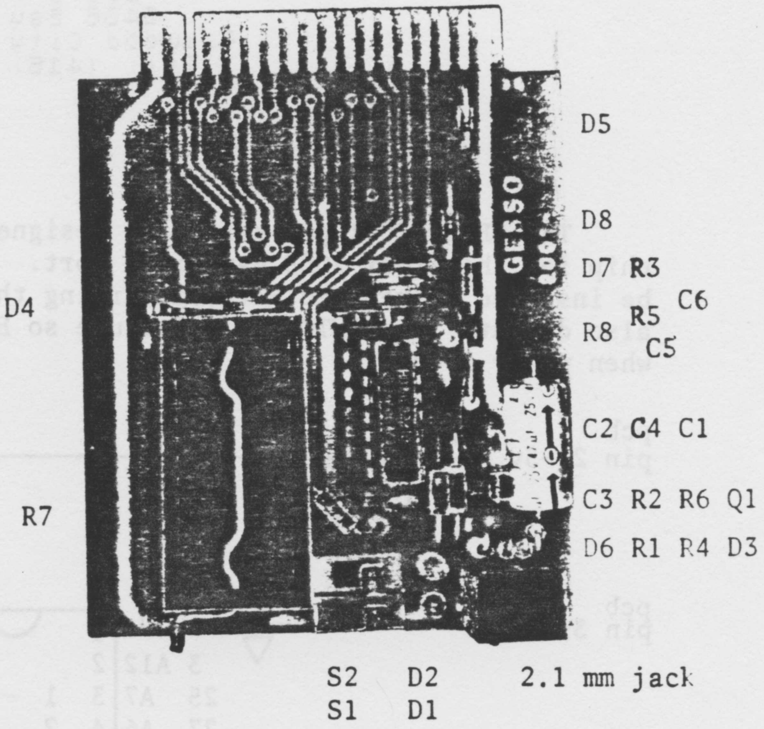


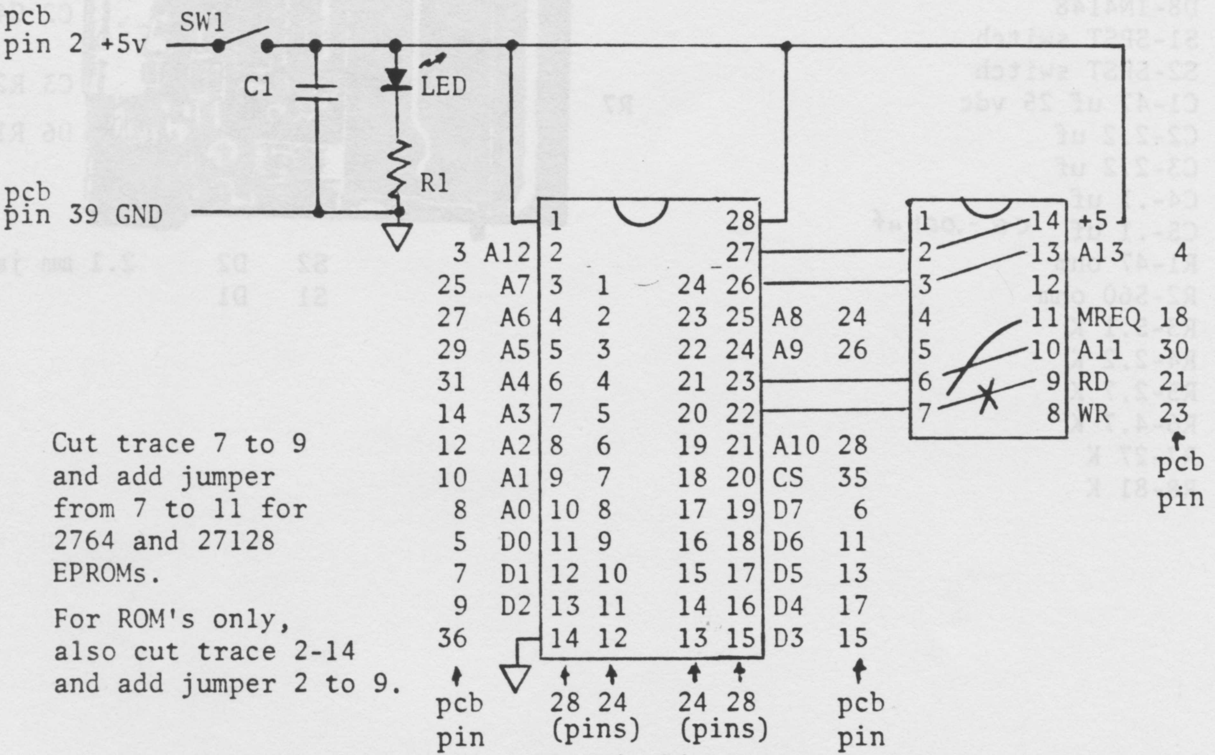
Figure 4-3. Parts List and Parts Locations

May 85

GESSO PRODUCTS

Bob Orrfelt
3436 Bay Road
Redwood City, CA 94063
Tel: (415) 369 9136

The GESSO 2001 pcb series is designed to hold one EPROM, ROM, or RAM chip for the Timex 2068 cartridge port. A power switch allows the pcb to be inserted or removed without turning the computer power off. This switch also defeats the AUTO start feature so EPROMs can be read. An LED shows when power is on.



Cut trace 7 to 9 and add jumper from 7 to 11 for 2764 and 27128 EPROMs.

For ROM's only, also cut trace 2-14 and add jumper 2 to 9.

Circuit Diagram.

For AUTO start EPROMS turn the computer power off and insert the pcb. Push the pcb switch to the left (on) and turn the computer on. If the computer has been used for other things, the NEW command will AUTO start.

To read an EPROM or ROM, use OUT 244,16 for 8K or OUT 244,48 for 16K.

To move the code from EPROM to RAM:

```
10 FOR n=32768-40959
20 OUT 244,16: LET x= PEEK n
30 OUT 244,0: POKE (n+16384),x
40 NEXT n
```

Figure 4-4. GESSO Cartridge Board Data

GESSO 2001 series:

2001.00 Bare board	$ 9.95
2001.11 Kit/zif	15.95
2001.12 Assembled/zif	24.95

An anti-static bag is supplied with each pc board. It has a slight pink color and should be used to store the board when an EPROM is installed in the socket and not in use.

The bare board requires a 28 pin socket, a capacitor, and at least a jumper in place of a switch. The LED and associated resistor are optional.

The switch is ALCO SLS 121-RA or equal. Push it tight against the board and solder both sides and it can be used as a handle.

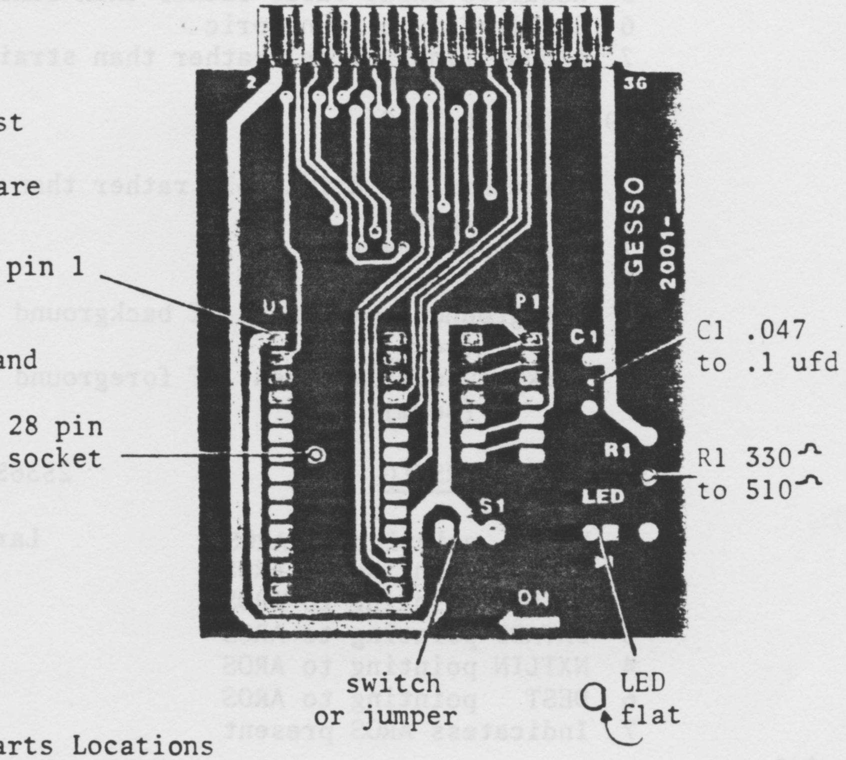


Figure 4-5. Parts Locations

TS2068 FLAGS

23611 5C3B FLAGS

0 Suppress space before tokens
1 Printing to printer, not TV
2 L MODE, not K, at current character
3 L MODE, not K, at cursor
5 Keyhit found
6 Numerical expression, not string
7 Interpret rather than CHECK SYNTAX

23612 5C3C TVFLAG

0 Printing to lower half of screen
1 Outputting line for edit or no. for string
3 Echo requested if inputting from keyboard
4 Outputting an automatic listing
5 Clear lower half when key pressed

23658 5C6A FLAGS2

0 Auto listing on screen
1 Printer buffer not empty
2 Inside string when doing KB in LISTCH
3 CAPS SHIFT LOCK on
4 Retype possible after syntax error
5 Delete Key Repeat (Key held down)

23665 5C71 FLAGX

0 Flexible length assignment required
1 Destination of assignment not found
5 Requires input value rather than line of program
6 Required type is numeric
7 Linput (Input Line) rather than straight input

23697 5C91 P FLAG

0 New chars XOR'd into old rather than being loaded
1 " (perm)
2 New char inverted
3 " (perm)
4 Foreground:=complement of background
5 " (perm)
6 Background:=complement of foreground
7 " (perm)

23750 5CC6 ARSFLG

0 X_PTR pointing to AROS
1 CH_ADD pointing to AROS
2 K_CUR pointing to AROS
3 DATADD pointing to AROS
4 NXTLIN pointing to AROS
6 DEST pointing to AROS
7 Indicatess AROS present

25365 6315 FLAGM

Largest BANK number

CHUNK SWITCHING CODES

Decimal Codes

Chunk	Addresses	OUT Code			DOCK Header		
		8K	16K	32K	8K	16K	32K
7	57344-65535	128	192	240	127	63	15
6	49152-57343	64			191		
5	40960-49151	32	48		223	207	
4	32768-40959	16			239		
3	24576-32767	8	12	15	247	243	240
2	16384-24575	4			251		
1	8192-16383	2	3		253	252	
0	0-8191	1			254		

Hex Codes

Chunk	Addresses	OUT Code			DOCK Header		
		8K	16K	32K	8K	16K	32K
7	E000-FFFF	80	C0	F0	7F	3F	0F
6	C000-DFFF	40			BF		
5	A000-BFFF	20	30		DF	CF	
4	8000-9FFF	10			EF		
3	6000-7FFF	08	0C	0F	F7	F3	F0
2	4000-5FFF	04			FB		
1	2000-3FFF	02	03		FD	FC	
0	0000-1FFF	01			FE		

from BASIC:

```
OUT 244,48
```

from machine code:

```
LD   A,#30
OUT  (F4),A
```

SYSTEM CONFIGURATION TABLE

Decimal	Hex	Byte	Comment
24298	5EEAh	01	BASIC
24299	5EEBh	02	AROS
24300	5EECh	08	LO Starting Address
24301	5EEDh	80	HI Starting Address
24302	5EEEh	EF	CHUNK 4
24303	5EEFh	01	AUTO start
24304	5EF0h	15	LO byte vars
24305	5EF1h	00	HI byte vars
24306	5EF2h	02	LROS
24307	5EF3h	08	LO Starting Address
24308	5EF4h	80	HI Starting Address
24309	5EF5h	EF	CHUNK 4
24310	5EF6h	80	End of Table

Bob Orrfelt, 3436 Bay Road, Redwood City, CA 94063. Tele: (415) 369 9136